Class 17

Isabella Ruud PID: A59016138

Table of contents

Stuff done in the terminal: 1

Downstream analysis 3

Remove zero count genes 4

Try a PCA 5

DESeq analysis 9

Stuff done in the terminal:

to connect to the AWS machine, can run:

ssh -i "bioinf_isabellaruud.pem" ubuntu@ec2-54-200-30-181.us-west-2.compute.amazonaws.com

set environment variable for key export KEY="~/Downloads/bioinf_isabellaruud.pem"

and can check that the variable was stored correctly using echo #KEY

set environment variable for the server export SERVER="ubuntu@ec2-54-200-30-181.us-west-2.compute.amazonaws.com"

now can run this to connect ssh -i $KEY $SERVER

Download Ubuntu binaries for SRA-toolkit curl -O https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit
ubuntu64.tar.gz

Unzip and Untar SRA-toolkit files gunzip sratoolkit.current-ubuntu64.tar.gz tar -xvf sratoolkit.current-ubuntu64.tar

or tar -zxvf sratoolkit.current-ubuntu64.tar.gz

add the path to sra toolkit in the environment path variable so that we don't have to type it in each time

export PATH=$PATH:/home/ubuntu/sratoolkit.3.2.0-ubuntu64/bin

download the files in the SRA ID SRR600956 prefetch SRR600956

reconstruct the fastq files using: fastq-dump SRR600956

can use head to look into the file head -8 SRR600956.fastq

to figure out how many reads are in the file: grep -c "@SRR600956" SRR600956.fastq

25849655 reads

can also check with: tail SRR600956.fastq

Now switching to RNA seq data:

use prefetch to load the SRR file prefetch SRR2156848

get the fastq files in paired end format fastq-dump –split-3 SRR2156848

check how many reads are in each file grep -c "@SRR2156848" SRR2156848_1.fastq 2959900 reads

grep -c "@SRR2156848" SRR2156848_2.fastq 2959900 reads

can do all at once with grep -c "@SRR" *.fastq

repeat this process of prefetch and fastq-dump for SRR2156849, SRR2156850, SRR2156851

prefetch SRR2156849 SRR2156850 SRR2156851

fastq-dump –split-3 SRR2156849 SRR2156850 SRR2156851

download kallisto wget https://github.com/pachterlab/kallisto/releases/download/v0.44.0/kallisto_linux-v0.44.0.tar.gz

unzip and untar gunzip kallisto_linux-v0.44.0.tar.gz tar -xvf kallisto_linux-v0.44.0.tar

add kallisto path to environment variable export PATH=$PATH:kallisto_linux-v0.44.0/

get human hg19 reference transcriptome

wget ftp://ftp.ensembl.org/pub/release-67/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh37.67.cdna.all.fa.gz

unzip the file gunzip Homo_sapiens.GRCh37.67.cdna.all.fa.gz

count up the number of sequences grep -c ">" Homo_sapiens.GRCh37.67.cdna.all.fa 176981 sequencings

grep -c "protein_coding" Homo_sapiens.GRCh37.67.cdna.all.fa 176981

build the transcriptome index using kallisto kallisto index -i hg19.ensembl Homo_sapiens.GRCh37.67.cdna.all.fa

Run transcript quantification for the pair of SRR2156848 FASTQ files: kallisto quant -i hg19.ensembl -o SRR2156848_quant SRR2156848_1.fastq SRR2156848_2.fastq

run for the other files kallisto quant -i hg19.ensembl -o SRR2156849_quant SRR2156849_1.fastq SRR2156849_2.fastq

kallisto quant -i hg19.ensembl -o SRR2156850_quant SRR2156850_1.fastq SRR2156850_2.fastq

kallisto quant -i hg19.ensembl -o SRR2156851_quant SRR2156851_1.fastq SRR2156851_2.fastq

or use a script to automate this using nano that has these lines of code

make that script executable chmod +x runme.sh

./runme.sh

get results back to computer scp -r -i "~/Downloads/bioinf_isabellaruud.pem" ubuntu@ec2-34-219-113-54.us-west-2.compute.amazonaws.com:~/*_quant .

Downstream analysis

```
folders <- list.files(pattern = "_quant")
files <- paste0(folders, "/abundance.h5")
files
```

```
[1] "SRR2156848_quant/abundance.h5" "SRR2156849_quant/abundance.h5"
[3] "SRR2156850_quant/abundance.h5" "SRR2156851_quant/abundance.h5"
```

```
file.exists(files)
```

```
[1] TRUE TRUE TRUE TRUE
```

load up tximport package

```
library(tximport)
```

have names 1,2,3,4 so don't know what samples they are so will add names

```
names(files) <- sub("_quant", "", folders)
```

```
txi.kallisto <- tximport(files, type = "kallisto", txOut = TRUE)
```

```
1 2 3 4
```

```
#head(txi.kallisto)
```

see how many reads are in each sample

```
colSums(txi.kallisto$counts)
```

```
SRR2156848 SRR2156849 SRR2156850 SRR2156851
   2563611    2600800    2372309    2111474
```

Remove zero count genes

Need to filter out annotated transcripts with no reads

```
to.keep <- rowSums(txi.kallisto$counts) > 0
kset.nonzero <- txi.kallisto$counts[to.keep,]
```

check how many genes are left

```
nrow(kset.nonzero)
```

```
[1] 94561
```

remove genes with no change over the samples

```
keep2 <- apply(kset.nonzero,1,sd)>0
x <- kset.nonzero[keep2,]
```

see how many are left

```
nrow(x)
```

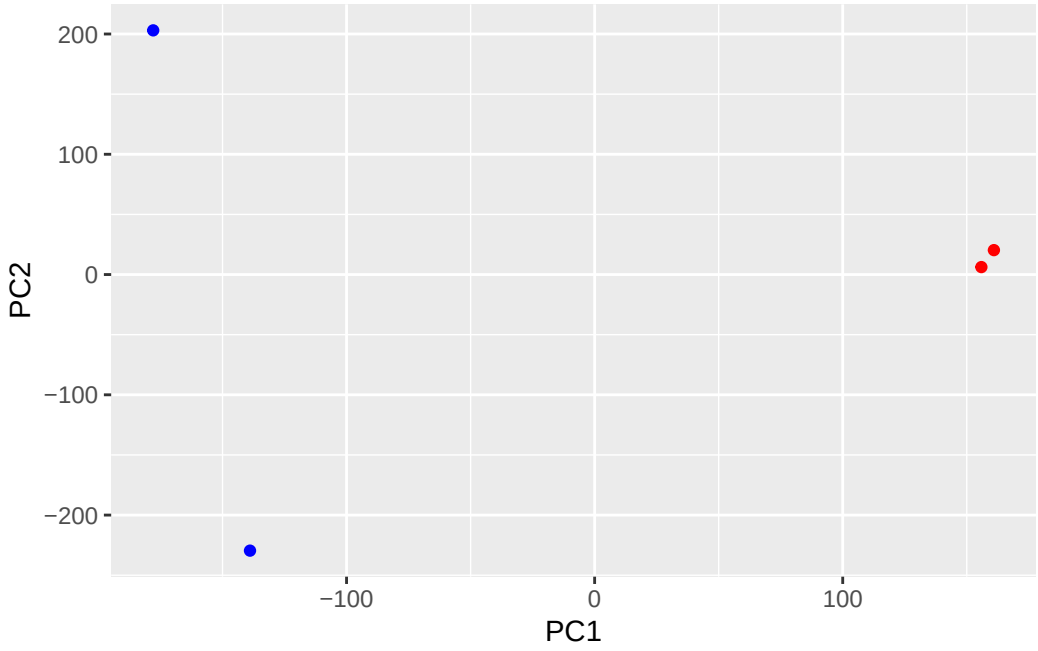
```
[1] 94525
```

Try a PCA

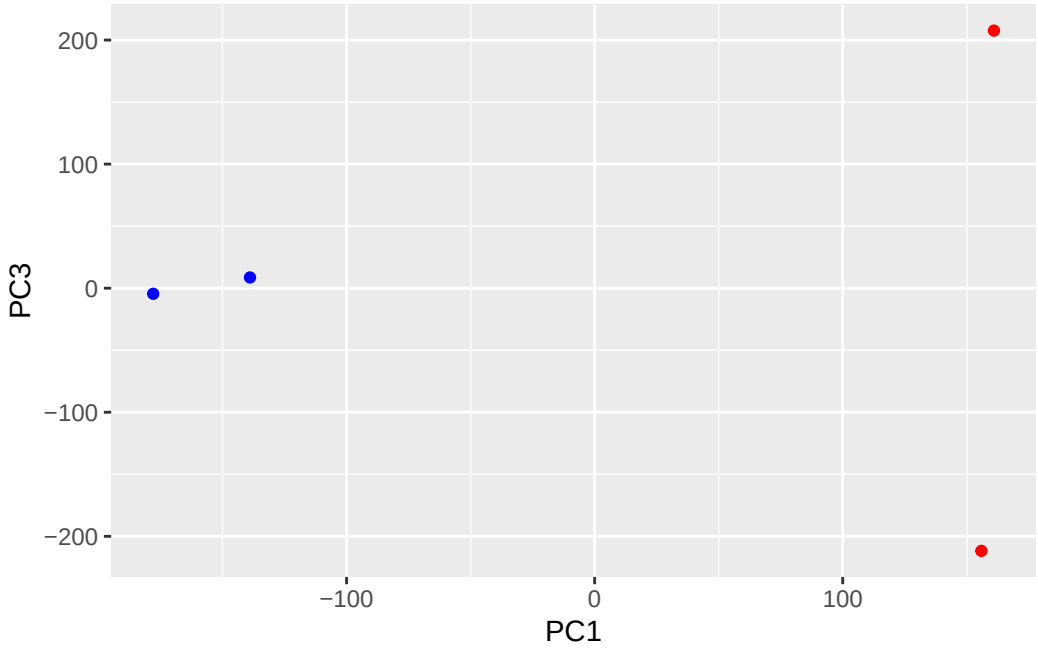
```
pca <- prcomp(t(x), scale = TRUE)
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3   PC4
Standard deviation     183.6379 177.3605 171.3020 1e+00
Proportion of Variance   0.3568   0.3328   0.3104 1e-05
Cumulative Proportion    0.3568   0.6895   1.0000 1e+00
```

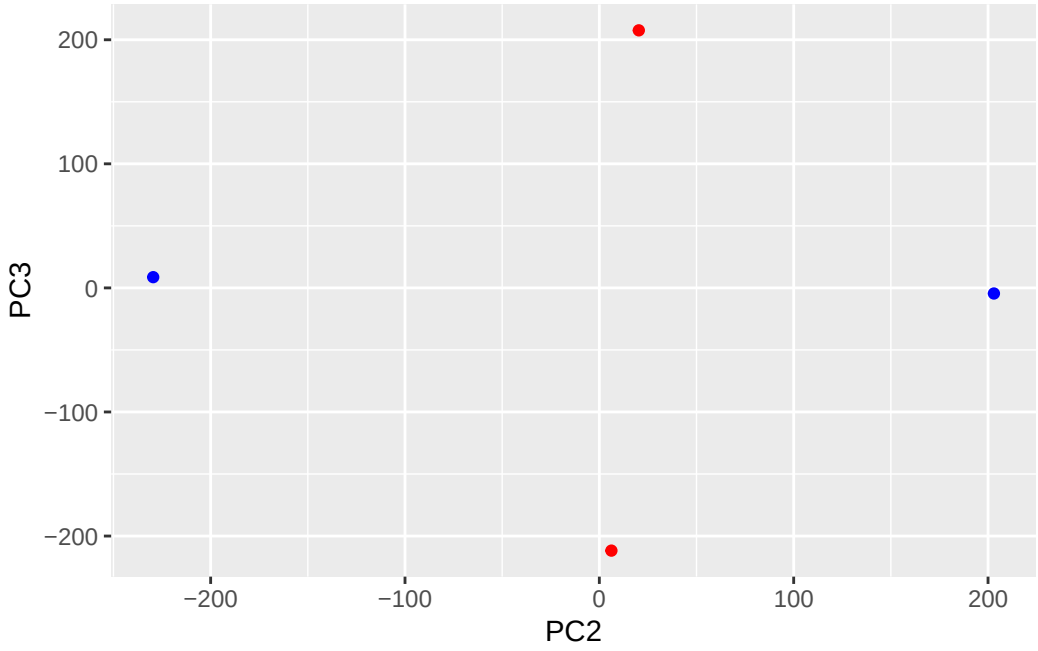
```
library(ggplot2)

ggplot(pca$x) +
  aes(PC1, PC2) +
  geom_point(col=c("blue","blue","red","red"))
```



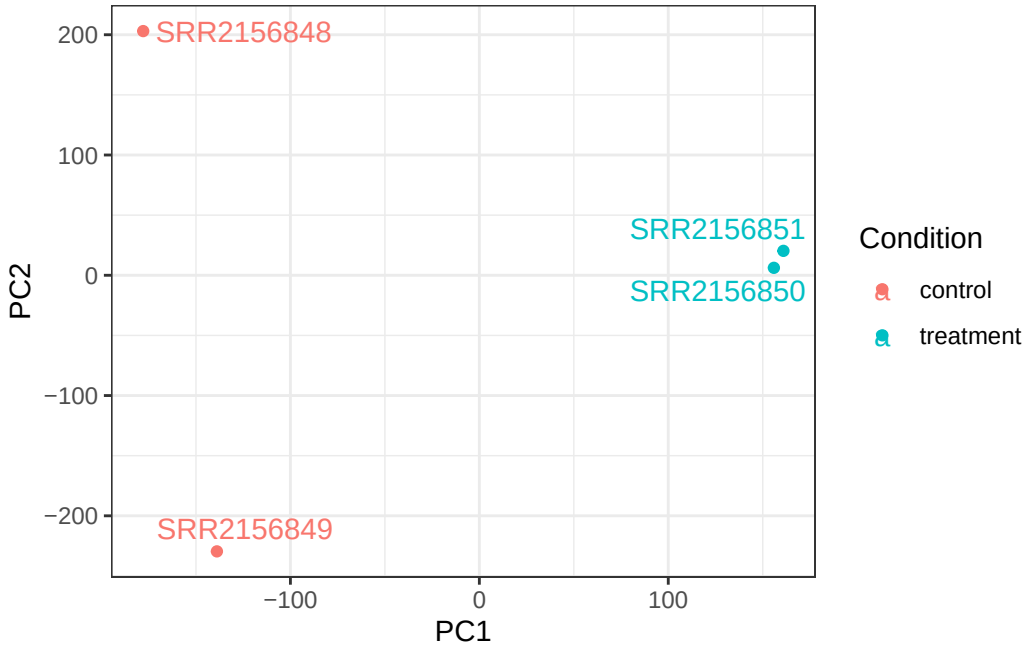

```
ggplot(pca$x) +
  aes(PC1, PC3) +
  geom_point(col=c("blue","blue","red","red"))
```




```
ggplot(pca$x) +
  aes(PC2, PC3) +
  geom_point(col=c("blue","blue","red","red"))
```

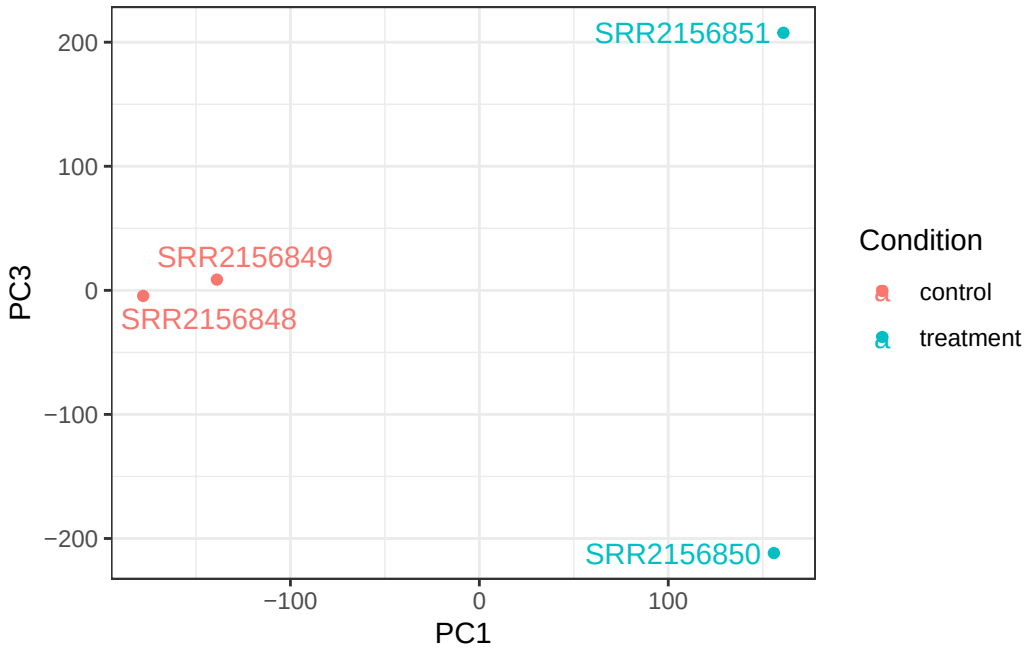


```
library(ggrepel)

colData <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(colData) <- colnames(txi.kallisto$counts)

y <- as.data.frame(pca$x)
y$Condition <- as.factor(colData$condition)

ggplot(y) +
  aes(PC1, PC2, col=Condition) +
  geom_point() +
  geom_text_repel(label=rownames(y)) +
  theme_bw()
```

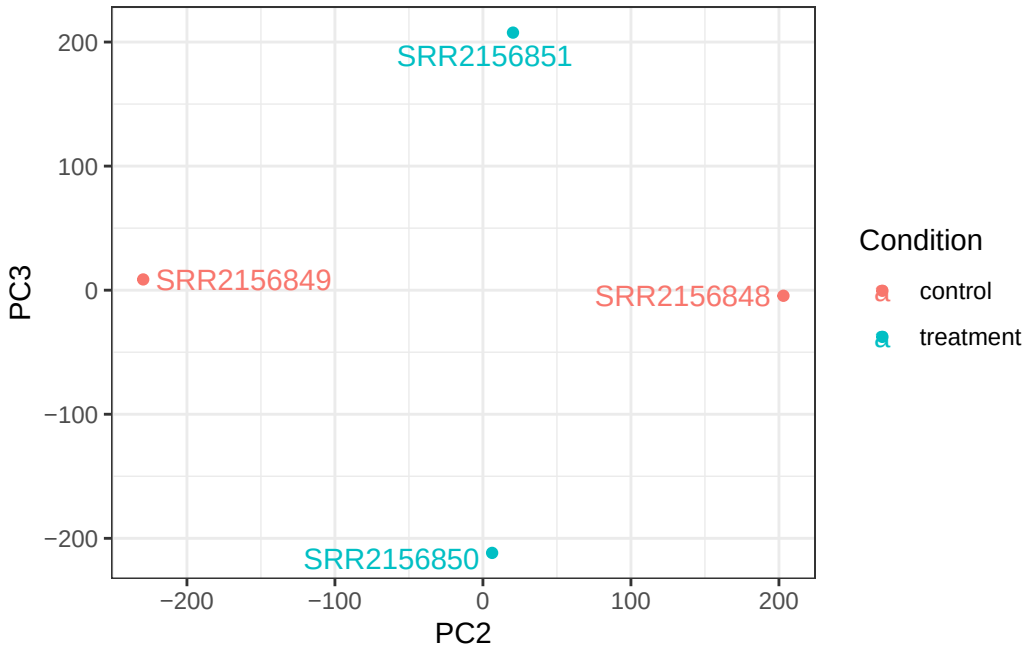



```
colData <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(colData) <- colnames(txi.kallisto$counts)

y <- as.data.frame(pca$x)
y$Condition <- as.factor(colData$condition)

ggplot(y) +
  aes(PC1, PC3, col=Condition) +
```

```
  geom_point() +
  geom_text_repel(label=rownames(y)) +
  theme_bw()
```




```
colData <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(colData) <- colnames(txi.kallisto$counts)

y <- as.data.frame(pca$x)
y$Condition <- as.factor(colData$condition)

ggplot(y) +
  aes(PC2, PC3, col=Condition) +
  geom_point() +
  geom_text_repel(label=rownames(y)) +
  theme_bw()
```



DESeq analysis

```
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```

Attaching package: 'BiocGenerics'
```

```
The following objects are masked from 'package:stats':
```

```
    IQR, mad, sd, var, xtabs
```

```
The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
    table, tapply, union, unique, unsplit, which.max, which.min


Attaching package: 'S4Vectors'

The following object is masked from 'package:utils':

    findMatches

The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges


Attaching package: 'IRanges'

The following object is masked from 'package:grDevices':

    windows

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats
```

```
Attaching package: 'MatrixGenerics'

The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

Loading required package: Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.


Attaching package: 'Biobase'

The following object is masked from 'package:MatrixGenerics':

    rowMedians

The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians
```

```
sampleTable <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(sampleTable) <- colnames(txi.kallisto$counts)
```

```
dds <- DESeqDataSetFromTximport(txi.kallisto,
                                sampleTable,
                                ~condition)
```

```
using counts and average transcript lengths from tximport
```

```
dds <- DESeq(dds)
```

```
estimating size factors
```

```
using 'avgTxLength' from assays(dds), correcting for library size
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
-- note: fitType='parametric', but the dispersion trend was not well captured by the
   function: y = a/x + b, and a local regression fit was automatically substituted.
   specify fitType='local' or 'mean' to avoid this message next time.
```

```
final dispersion estimates
```

```
fitting model and testing
```

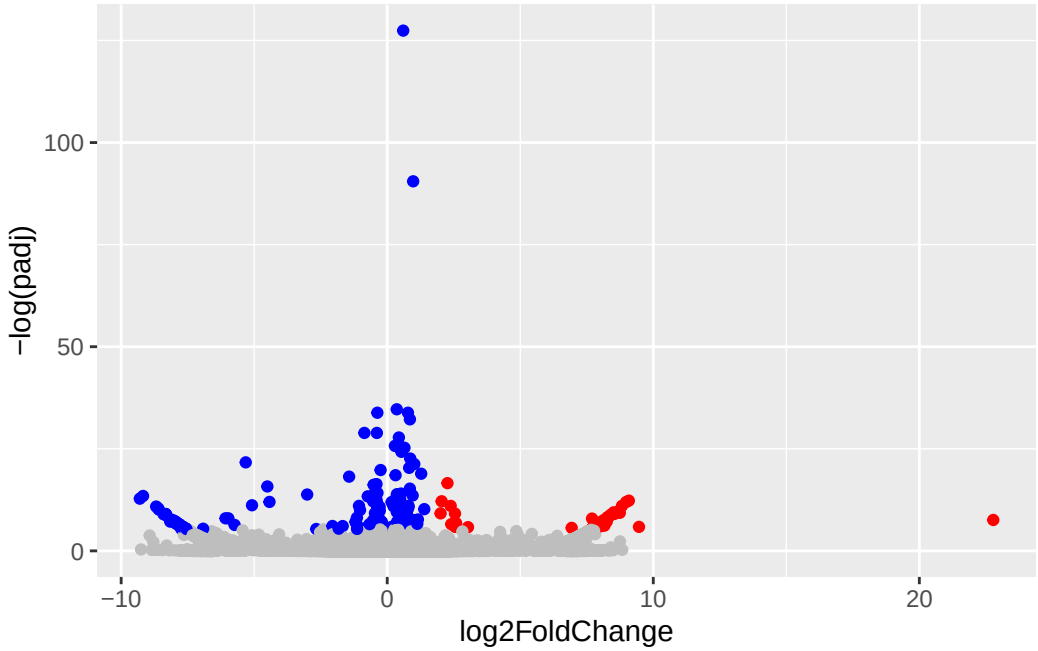
```
res <- results(dds)
head(res)
```

```
log2 fold change (MLE): condition treatment vs control
Wald test p-value: condition treatment vs control
DataFrame with 6 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
```

```
ENST00000539570  0.000000             NA        NA        NA        NA
ENST00000576455  0.761453       3.155061   4.86052 0.6491203  0.516261
ENST00000510508  0.000000             NA        NA        NA        NA
ENST00000474471  0.484938       0.181923   4.24871 0.0428185  0.965846
ENST00000381700  0.000000             NA        NA        NA        NA
ENST00000445946  0.000000             NA        NA        NA        NA
                      padj
                 <numeric>
ENST00000539570         NA
ENST00000576455         NA
ENST00000510508         NA
ENST00000474471         NA
ENST00000381700         NA
ENST00000445946         NA
```

```
mycol <- c(rep("grey", nrow(res)))
mycol[res$log2FoldChange > 2] <- "red"
mycol[res$log2FoldChange < 2] <- "blue"
mycol[res$padj > 0.005 ] <- "grey"
ggplot(res) +
  aes(x=log2FoldChange, y = -log(padj)) +
  geom_point(col=mycol)
```

```
Warning: Removed 147246 rows containing missing values or values outside the scale range
(`geom_point()`).
```


100
50
0
−log(padj)
−10
0
10
20
log2FoldChange